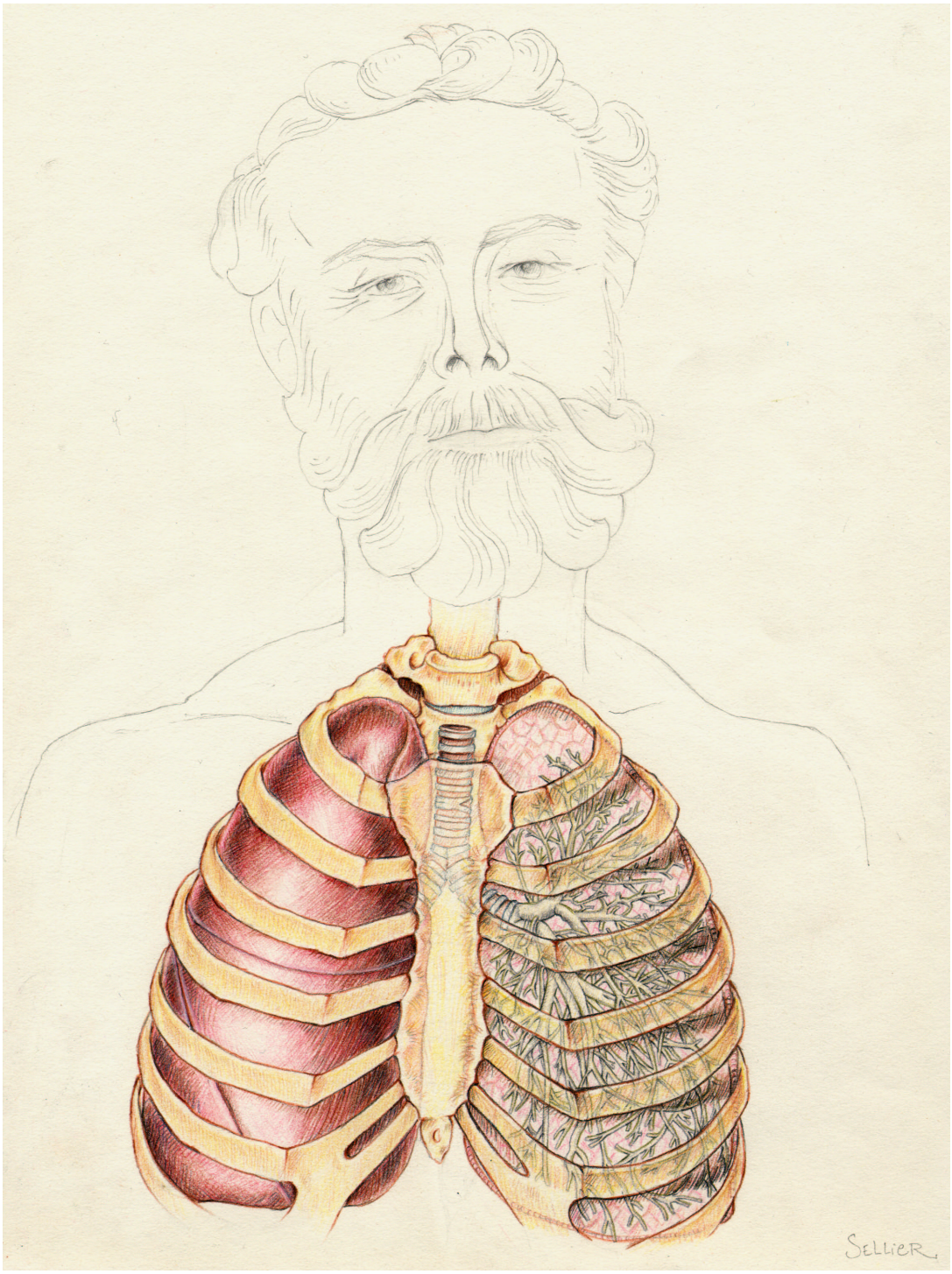
III

Visualisation et Quantification de Structures Anatomiques Arborescentes


SELLIER

Chapter 7

Outils de Visualisation et de Mesure

Résumé — Dans ce chapitre, nous présentons différents outils nécessaires pour la visualisation et la quantification de pathologies dans les objets segmentés. Comme nous l'avons déjà fait dans le cadre d'applications comme l'extraction de surfaces d'anévrismes dans des images 3DRA, ou bien la visualisation des polypes à la surface du colon, dans le chapitre 6, notre outil de segmentation de surface basé sur les ensembles de niveaux et le *Fast-Marching* permet d'extraire et de représenter des objets en 3D.
En premier lieu nous parlons brièvement des problèmes soulevés par la visualisation de surfaces implicites en 3D, et en particulier des spécificités des surfaces définies par les *Level-Sets* dans la section 7.1.
En s'appuyant sur un ensemble de trajectoires qui décrivent nos surfaces - comme le squelette dans le cas de structures arborescentes - nous développons ici des outils de mesure et d'observation des pathologies. Notamment, nous nous intéressons à la mesure du volume en section 7.1, et à la mesure de sections de nos objets segmentés en section 7.2. Ces outils seront utilisés dans toute la suite de cette partie.

Abstract — In this chapter we introduce the different necessary tools for visualization and quantification of our segmented objects. The final result of a segmentation, given by our framework as done in chapter 6 for different applications, can lead to visualization and measures on the global object.
We first briefly present the problems of visualization of an implicit surface in 3D, and more precisely the specific drawbacks of the *Level-Sets* representation in section 7.1. Assuming that we can extract a whole set of trajectories in a tree-shaped object, we present the different tools that will measure the pathologies, on the basis of those trajectories. Important measurements include: volume measurements, as explained in section 7.1, and objects cross-section measurements, as detailed in section 7.2. Those tools are useful for the framework developed in the following chapters.

7.1 Visualization of 3D segmentation

In the domain of medical image analysis, the segmentation tools we developed are essentially interesting when applied to 3D images. This is the reason why the implementation we build is designed for this image dimensionality, and that our visualization efforts were mainly directed to 3D techniques.

In this section the reader will first find a short presentation of the basic notions of virtual reality needed to understand the content of our work. They are grouped together in the first subsection, which can be skipped by the readers who are already familiar with them.

7.1.1 Virtual reality notions

Classically, the basic techniques for computer graphics of virtual reality rely on the computation of *renderings* of virtual 3D *scenes*. A scene is composed of virtual *actors*, *lights* and a *camera*.

What are actors ?

The term *actor* covers everything that might be seen when properly enlighten. For instance, in a virtual reality model of a house, each piece of furniture would be modeled by a specific actor, and so would be the floors, walls, stairs, etc.

Traditionally, the shape of a 3D actor is explicitly modeled by a set of graphic primitives: points, lines and surface patches. In recent and advanced models, the surface of an actor is sometimes modeled using implicit functions.

According to the complexity of the modelization, the rendered appearance of the surface of an actor can depend on many and various parameters:

- the position and orientation of the camera relatively to the actor surface;
- the properties of the surface which are taken into account by the *illumination model*;
- the positions, orientations, colors and attenuation factors of the lights, which can be at finite distance (punctual lights) or infinite distance;
- the positions and orientations of the other actors which may cause occlusions, projected shades, or even reflect light sources in advanced models.

What is an illumination model ?

The illumination model is the set of equations used to compute the color and brightness of a point on the surface of an actor according to:

- the angles of incidence, intensities and colors of the incident rays of light;
- the modeled properties of the surface;
- the angle of the departing ray of light.

The surface properties usually include colors, an opacity factor, reflectance, a specular power parameter, etc...

In addition to the illumination model, a *shading* model can be used to avoid the faceted aspect of polygonal surfaces. The most popular shading models are Gouraud and Phong shadings, although Phong shading is rarely used because of its computational cost.

What is the exact role of the camera ?

The camera plays the same role as in the making of a movie: the rays of light which encounter the objective determine the rendered (i.e. virtually acquired) 2D image. The usual parameters of a camera are its position, orientation, and two of the following parameters: view angle, focal distance and image size. The rendered 2D image is a projection of the illuminated actors in the focal plane of the camera.

Object-based and image-based rendering

The construction of the 2D image acquired by the camera may be *object-based* or *image-based*. In the case of object-based rendering, the actors are rendered one by one by applying the illumination model and the projection equations to the graphic primitives they are composed of. The occlusions are generally dealt with using a so-called *Z-buffering* technique: the final image is the result of the superposition of layers which correspond to different depths (Z-coordinate) in the scene. The points that are the closest to the camera are visible, others are more or less occluded according to the opacity of the points that are in front of them.

Object-based rendering is not a recent technique, but it is fast, rather simple, and can be accelerated by specialized hardware devices. For example, OpenGL hardware implementations make interactive renderings of simple scenes possible even on a low end PC. The main drawbacks of object-based rendering are that photo-realistic images are difficult to achieve, especially in the case of complex scenes, and that multiple reflections are usually not taken into account. Moreover the actors have to be explicitly represented using graphic primitives.

In the case of image-based rendering (or *ray-tracing*) the color and brightness of each point of the rendered image is computed by tracing a ray starting from this point. The illumination model is invoqued when the ray hits an actor, and reflections on several actors are even possible before reaching a light source. In the most advanced computer graphics software products based on ray-tracing, the actors can also have implicit representations.

The images produced by ray-tracing can be of very high quality, but the major drawback of image-based rendering is the computation cost related to the calculation of the rays.

7.1.2 Visualization of a level-set

Visualizing a level set is nothing more than visualizing an iso-surface in 3D, or an iso-contour in 2D. More generally the hypersurface which needs to be visualized is

the zero-level set of $\phi(\cdot, t)$, where t is fixed, which is a function defined over the image domain $\Omega \subset \mathbb{R}^3$ in our applications. In figure 7.1-left, the surface of a sphere is implicitly defined by the signed distance to itself.

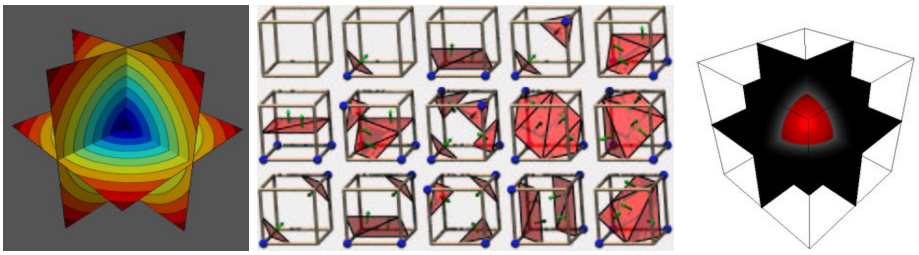


Figure 7.1. The marching cubes algorithm: Left image represents the iso-values of the signed distance to a sphere in 3D; middle image represents the different configuration encountered by the *Marching-Cubes* ; right image is a smooth surface rendering of the triangles that approximate the implicitly defined sphere of figure 7.1-left given by the algorithm.

The 3D visualization of the zero level-set surface by object-based rendering algorithms cannot be done directly. An explicit representation of the surface by polygonal graphic primitives has to be computed first.

Several approaches are possible for the computation of a polygonal approximation to an iso-surface. The most popular of all is certainly the *Marching-Cubes* (see [104]), which computes a triangulated surface. In each cube formed by eight contiguous voxel centers, the values of the implicit function at the vertices of the cube are compared to the specified iso-value. The possible configurations are classified (see figure 7.1-middle), and a look-up table is used to quickly give a triangulated approximation of the intersection of the iso-surface with the currently examined cube. All the cubes are examined one by one in a raster-scan "marching" fashion, in opposition to the algorithms which try to "track" the iso-surface.

But sometimes the *Marching-Cubes* algorithm generates triangle sets containing holes, due to ambiguous cases. Many authors have proposed solutions, for example the marching tetrahedra algorithm in [166].

However, we chose the *Marching-Cubes* for reasons of accuracy, reliability, and (above all) simplicity of use since efficient implementations of it are available. It provides an accurate triangulated surface whose precision leads to high-quality renderings, like in the endoscopic images shown in figure 7.2.

7.1.3 Problem with the *Marching-Cubes*

A classical evolution equation defined by $\frac{\partial \phi}{\partial t} + V \,.\, \nabla \phi = 0$ makes no distinction between the level sets of ϕ. They are all attracted by the same asymptotic hypersurface provided that they are sufficiently "close" to it. As a result, ϕ gets very steep in its vicinity, which causes the *Marching-Cubes* to give poor and aliased results.

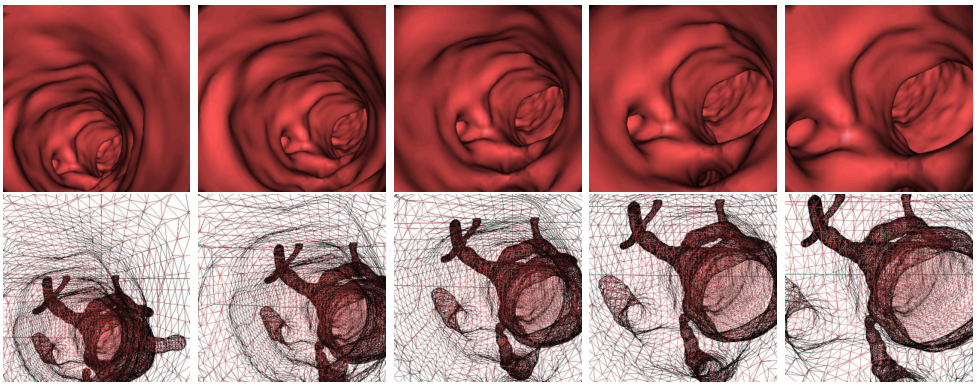


Figure 7.2. Surface rendering in the aorta: First row shows frames of an endoscopic movie in an aorta MR dataset. Second row is the wire-frame version of this movie, given by the *Marching-Cubes* where we can see the whole anatomical object and its several branches by transparency

In fact, the level-sets do not remain a distance function in many cases (an exception is a constant advection flow, for example see figure 4.6). This property at initialization, is lost after several iterations. Figure 7.3 shows two examples: the first one follows a balloon forces, which is positive inside a circle, and negative outside; the second one is a flow composed of a positive balloon force and a boundary based force, which stops the level sets of ϕ.

7.1.4 Restoring the distance function

In conclusion, the solution to the classical Hamilton-Jacobi evolution equation proposed in [135] is not a distance function. But this property is the hypothesis of several numerical techniques to accelerate convergence, like the fast geodesic active contours proposed in [65] and [67]. Moreover, the practical application of the level-set method is plagued with such questions as: when do we have to "reinitialize" the distance function? How do we reinitialize" the distance function. In [163], the author suggests that when the zero-level set evolves in the vicinity of the borders of the narrow-band, the distance to the zero-level set must be re-initialized. For the authors of [68, 69], this problem reveals a disagreement between the theory and its implementation, the authors propose an alternative to the use of Hamilton-Jacobi evolution equation which eliminates this contradiction. In order to reach this goal, they look for a function $B : \mathbb{R}^3 \times \mathbb{R}^+ \rightarrow \mathbb{R}$ such that $\frac{\partial \phi}{\partial t} = B$ and which satisfies the two constraints

- ϕ is a distance function
- $\frac{\partial \phi}{\partial t} = \beta \mathcal{N}$ where β is the velocity, and $\mathcal{N}$ the inward unit normal.

Those constraints lead to the new relation $\nabla \phi \cdot \nabla B = 0$. This efficient method increases the computing cost.

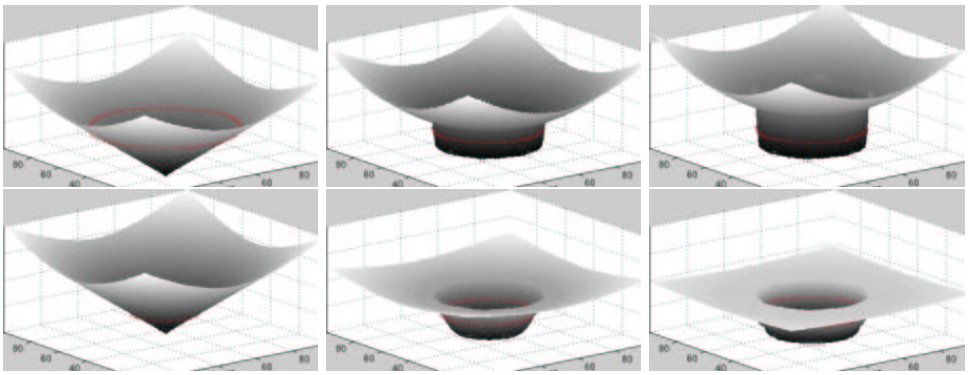


Figure 7.3. Loosing the distance function when converging: First row shows consecutive iterations of the level-sets of a geodesic active contour that minimizes the distance to a circle; second row shows consecutive iterations of the level-sets of a geodesic active contour that inflates according to a balloon force with boundary based forces on the same circle.

Following the author of [84], we include the a restoration force in the Hamilton-Jacobi flows, which not only ensures the evolution of $\phi(0,t)$ given by equation (4.6), but also prevents $\phi(\cdot,t)$ from getting too steep in the vicinity of $\phi(0,t)$. This new partial differential equation is given by:

$$\frac{\partial \phi}{\partial t} + V \,.\, \nabla\phi = \mu \,.\, \mathrm{sgn}_\theta(\phi(x,t)) \,.\, (1 - \|\nabla\phi\|) \tag{7.1}$$

where V is the flow defined by equation (4.6) and where the modified signed function sgn_θ is defined by:

$$\mathrm{sgn}_\theta(y) = \begin{cases} -1 & \text{if} \quad y < -\theta \\ \frac{y}{\theta} & \text{if} \quad -\theta \leq y \leq \theta \\ 1 & \text{if} \quad \theta < y \end{cases}$$

The new differential operator introduced in equation (7.1) is inspired from the distance function restoration operator used in [170]. The modification of the sign function avoids the apparition of oscillations during the numerical approximation of equation (7.1), without having to introduce numerical flux or slope bounds. Otherwise, as signaled in [84, page 56], these oscillations are responsible for short but annoying displacements of the zero-level set of $\phi(\cdot,t)$. And the author of [84] proposes to use another scheme, originally presented in [159], which inflates and deflates successively the level-set in order to extract the distance to the zero level-set, without displacing it. We choose not to add another bunch of computations to our method, and decide to use method of [170].

The parameter denoted by θ can be set to a fixed value (we used $\theta = 10$ in our experiments). The parameter μ defines the weight of the newly introduced differential operator, and has to be adapted according to the other forces parameters. If μ is too small, then $\phi(\cdot,t)$ is likely to get too steep for the *Marching-Cubes* to give good

results. But too high values of μ will increase the global CFL number, and thus cause the convergence of ϕ to be slower. In practice, it is not difficult to find a good value for μ.

The use of this new equation (7.1) is illustrated in figure 7.4, where the flow drives the zero level-sets to a sphere.

Figure 7.4. Aliasing when converging: Left image is the surface extracted at convergence when the level set matches the surface; Right image is the same result including a force to restore the distance function.

7.1.5 Volume versus Surface Rendering

Volume rendering is an advanced image-based visualization technique based on the integration of a transfer function along rays cast in a dense volume (i.e. a 3D image). The transfer function is generally based on the intensity and gradient of the image, and gives an opacity value for each voxel of the image. Surface (see figure 7.5-left) versus Volume (see figure 7.5-right) rendering is still an open question, and the choice between those two methods depends on the application.

With the shape extraction techniques we use, surface rendering has several advantages:

1. with the segmentation framework we have developed, the visualization of the anatomical object with surface based rendering does not need any input, any interaction (unless the color of the surface can be considered as an important parameter);

2. parameterization-free means robustness. Volume-based rendering relies on the critical choice of a suitable transfer function. Surface-based rendering is the direct representation of the surface extracted by the segmentation whereas the volume-based rendering relies on the user perception of the dataset;

3. Surface rendering is fast: when the triangulation has been extracted with the *Marching-Cubes* , endoscopic fly-through, like in figure 7.2 are generated in real-time, and OpenGL hardware implementations, now available on any low end PC, accelerate the computations. The computational cost of volume rendering

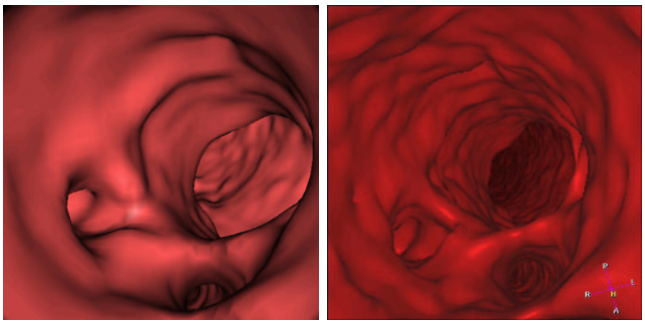


Figure 7.5. Comparing surface and volume rendering in the aorta: Left image is a surface rendered view generated with the *Marching-Cubes* on the final segmentation obtained; right image is a threshold based volume rendering view at the same location in the dataset.

is very high, and special hardware devices that might overcome this lack of performance are still under development.

Moreover, several artifacts may occur when using *Volume Rendering* on volume data (among them, aliasing, stair-casing and slicing, see [146]).

For all those reasons, we used the surface-based rendering for visualization, as well for inspection of results, as for endoscopic viewings. Notice that if surface-based rendering is parameter-free, it critically relies on the result of the segmentation.

7.2 Measurement Tools

The main target of our path and shape extraction framework is to measure pathologies in tube-shaped objects, like aneurysms in brain vessels, and polyps in the colon. We detail in this section the different tools used for quantification of those pathologies, that are characterized by their sections and volumes. Extracting the shapes of our objects, with the *Marching-Cubes* [104], we use a consequence of the Gauss theorem, discretized on the vertices of the triangulation obtained.

7.2.1 Gauss Theorem

As classically [9], volume and section measurements are based on Gauss theorem:

Theorem 7.1 (Gauss) *Let Ω be a subset of $\mathbb{R}^d$, let its boundary Σ be a closed surface, and U a differentiable vector field, then:*

$$\int_{\Omega} divU\, dx = \oint_{\Sigma} U.N\, d\sigma$$

where N is the outward normal to Σ.

A consequence of Gauss theorem is that the volume $\mathcal{V}(\Omega)$ of Ω can be simplified as an integral over the boundary Γ

$$\mathcal{V}(\Omega) = \int_{\Omega} dx = \frac{1}{3} \int_{\Omega} \operatorname{div}(x)\, dx = \frac{1}{3} \oint_{\Gamma} x.N\, d\sigma. \tag{7.2}$$

7.2.2 Volume Measurement

We assume that a 3D tubular structure has been segmented with a level set method and that $\phi(\cdot, t)$ is known at convergence and denoted by $\tilde{\phi}$. We also assume that centered paths have been computed in the tubular structure.

The volume to be measured is defined by the user who chooses a path and specifies two points p_1 and p_2 on this path. The computed volume is the volume of the interior region of the tubular structure limited by the two plane section S_1 and S_2 associated to (p_1, Π_1) and (p_2, Π_2) and defined by $S_i = \Pi_i \cap \tilde{\phi}^{-1}(\mathbb{R}^-)$ $i = 1, 2$. Here is a step-by-step summary of our algorithm, which is illustrated by figure 7.6.

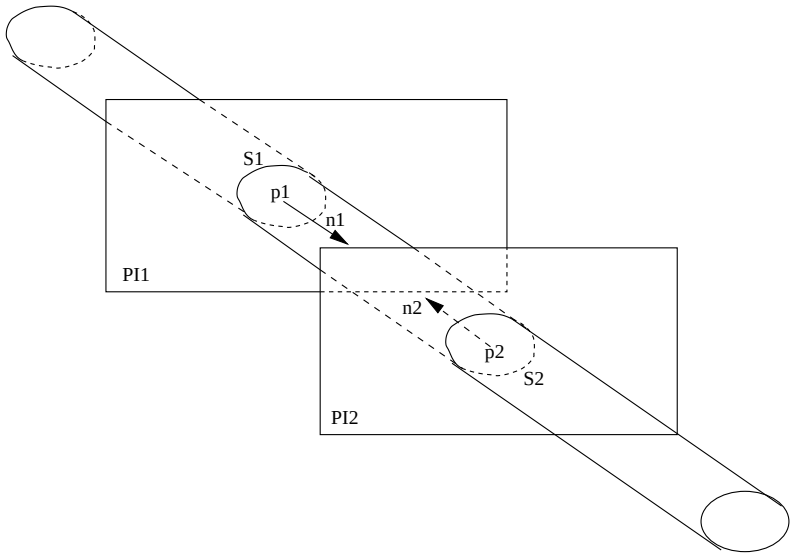


Figure 7.6. Volume measurement diagram.

- we compute tangent vectors to the path at p_1 and p_2, which are the normal vectors $\overrightarrow{n_1}$ and $\overrightarrow{n_2}$ to the plane sections S_1 and S_2;
- the equations $(p_1, \overrightarrow{n_1})$ and $(p_2, \overrightarrow{n_2})$ of the plane sections S_1 and S_2 are considered;
- the region of interest is actually the intersection of three subsets of $\mathbb{R}^3$, which are $\tilde{\phi}^{-1}(\mathbb{R}^-)$ and two half-spaces limited by the plane sections;
- we deduce the signed distance functions Ψ_1 and Ψ_2 to the two half-spaces Π_1 and Π_2 from the equations of the plane sections

- Considering that the shape of the object of interest can be complex, and lead to problems of intersection between planes Π_1 and Π_2 (see figure 7.8), we define a function Ψ the following way
  1. it is initialized with $\Psi(x, y, z) = \sqrt{3}, \forall (x, y, z)$ in the image domain;
  2. starting from the path point p_1 (respectively p_2), we apply a region growing algorithm, that labels only the voxels v which have $|\Psi_1(v)| < \sqrt{3}$, (respectively $|\Psi_2(v)| < \sqrt{3}$) and for those voxels, we set $\Psi(v) = \Psi_1(v)$ (respectively $\Psi(v) = \Psi_2(v)$);
  3. starting from any path point p not labeled between p_1 and p_2, we apply a connectivity filter that visits only the voxels where $\tilde{\phi}(v) < \sqrt{3}$ and that are not already visited by the first connectivity filter; and for each voxel visited we set $\Psi(v) = \tilde{\phi}(v)$; it enables to avoid including the interior of undesired parallel structures in the region of interest;
- the region of interest is equal to $\Psi^{-1}(\mathbb{R}^-)$, and a polygonal approximation of its boundary is computed by extracting the zero-level set of Ψ;
- the volume of the region of interest is computed using the following decomposition of equation (7.2) on the polygons of the extracted surface:

$$\mathcal{V}(\Psi^{-1}(\mathbb{R}^-)) = \frac{1}{3} \sum_{\mathrm{i}} \mathrm{g_i}.\,\mathrm{N_i}\,\sigma(\mathrm{P_i})$$

  where g_i, N_i and $\sigma(P_i)$ respectively denote the center of gravity, the outward normal and the surface of the polygon P_i.

The overall computation times are very short (less than 3 seconds for a $256 \times 256 \times 60$ image on a SunBlade 100), and the results on basic geometric primitives are excellent in terms of accuracy.

7.2.3 Example of volume measurement: an aneurysm

In this case, shown in figure 7.7, where the problem studied is the cerebral aneurysm of figure 6.4, the measurement of the aneurysm volume is done using one trajectory extracted inside a mask defined by the segmentation obtained in figure6.6. Taking two positions along the trajectory, we can easily define a volume of interest that contains the aneurysm. The volume shown in figure 7.7-right is not restricted to the aneurysm itself, and contain the surrounding vessel. But a good approximation can be given, by subtracting an approximate vessel volume, using the surfaces of the sections S_1 and S_2. Advantage of using our connectivity algorithm to obtain $\Psi(\mathbb{R}^-)$ instead of taking the region delimited by $\tilde{\phi}^{-1}(\mathbb{R}^-)$ and the two half space Π_1 and Π_2 determined by the distance functions Ψ_1, Ψ_2 is illustrated by figure 7.8 on the same dataset.

7.2.4 Section Measurement

We can also apply equation (7.2) in 2D to evaluate the surface limited by a closed planar curve. In order to illustrate this method, we show its application to a phantom dataset.

The data, shown in figure 7.9-left is the acquisition of a cube of Perspex (a type of plastic) with an aluminum rod in it, inside a dead human head. It was acquired

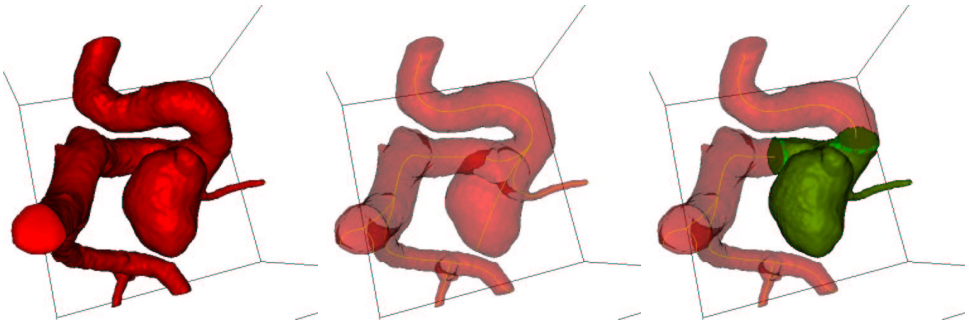


Figure 7.7. Measuring the volume of an aneurysm: The dataset used for this segmentation is shown in figure 6.4. Left image is the segmented object obtained in figure 6.6 by combining *Fast-Marching* and *Level-Sets* methods; middle image shows the multiple paths extracted; right image shows a sub-volume of the aneurysm which has been isolated.

with the Philips Integris **3D-RA** system. A **MIP** view in figure 7.9-right enables to see the variable section of the aluminum rod.

Following the results of chapter 6, we first segment the phantom with the *Fast-Marching* algorithm, starting from one point at the top of the aluminum rod. Computing the Euclidean path length while propagating, as detailed in section 2.2.3, it is very easy to extract the largest centered path, using the method described in section 2.3, with the thresholded distance $\tilde{\mathcal{D}}$ to the object borders. This path extracted is visible in figure 7.10-middle, by transparency. In a few iterations, the *Level-Sets* algorithm, with region-based forces, gives the result shown in figure 7.10-left.

In the experimental tool we built, the user specifies a particular path and obtains the section of the tubular structure according to the length of the path. The path is supposed to have a discrete representation, i.e. is represented by a list of points. Here we give a summary of the performed calculations for each point of the path:

- the normal of the section plane is computed using an approximation of the tangent vector to the path;
- an orthonormal base of the section plane is deduced;
- a rectilinear 2D grid, centered on the current path point, is defined on the section plane;
- at the center of each cell of the grid, the value of $\tilde{\phi}$ is computed by interpolation;
- an adequate algorithm is used (we used the Marching Squares) to compute an approximation of the zero iso-contour in the 2D grid;
- the surface enclosed in the resulting polygonal line, which in our example is drawn on the surface in figure 7.10-right, is computed thanks to a decomposition of equation (7.2) on the polygonal line.

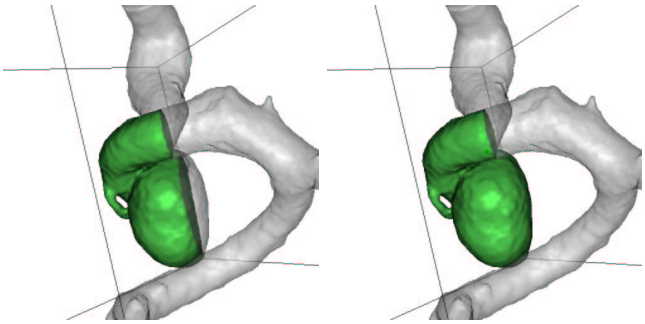


Figure 7.8. Advantage of the connectivity algorithm: left image shows the result of computing the intersection of $\tilde{\phi}^{-1}(\mathbb{R}^-)$, $\Psi_1^{-1}(\mathbb{R}^-)$, $\Psi_2^{-1}(\mathbb{R}^-)$. Right image is the representation of $\Psi(\mathbb{R}^-)$ superimposed on the segmentation along the same trajectory, between the same extremities.

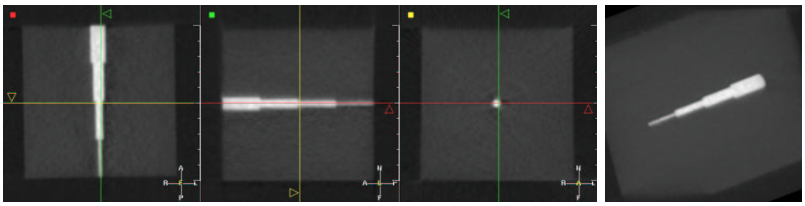


Figure 7.9. 3D-RA Phantom: On the left image are shown three orthogonal views of the perspex cube acquired with a **3D-RA** system; right image is a **MIP** view of this data-set.

Like in the case of volume measurements, the computation times are very short, and the algorithm gives very accurate measurements of basic geometric primitives. Concerning the phantom problem, we have computed this section at each path point (see figure 7.11). Figure 7.11 shows the measures done along the path displayed in figure 7.10-right. On the graphic, we have displayed the several real dimensions of the aluminum cylinders, and we have also displayed the interval of deviation of 2% that was indicated by a study on the accuracy of the calibration, the distortion correction, and the reconstruction of the **3D-RA** system [83]. The section measurements of the segmented object show that our method gives results which lie in those intervals, when the radius is more than one millimeter.

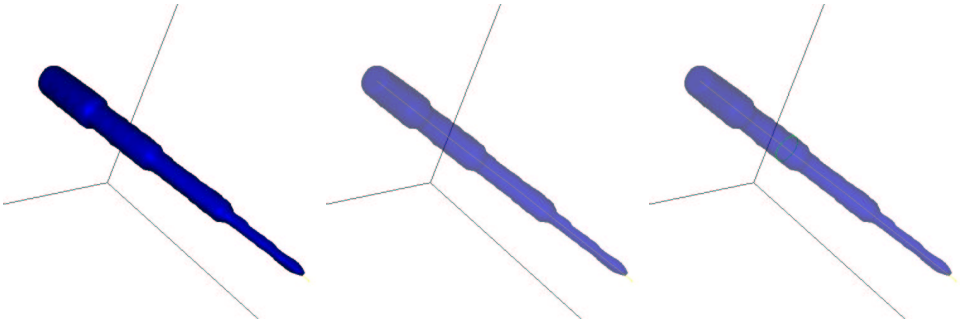


Figure 7.10. Segmentation result on the 3D-RA Phantom of figure 7.9: Left image is the segmented object with the combination of *Fast-Marching* and *Level-Sets* methods; middle image is the same object with opacity $< 1.$ and the path extracted; right image shows the intersection of the phantom surface with the section plan for measurements.

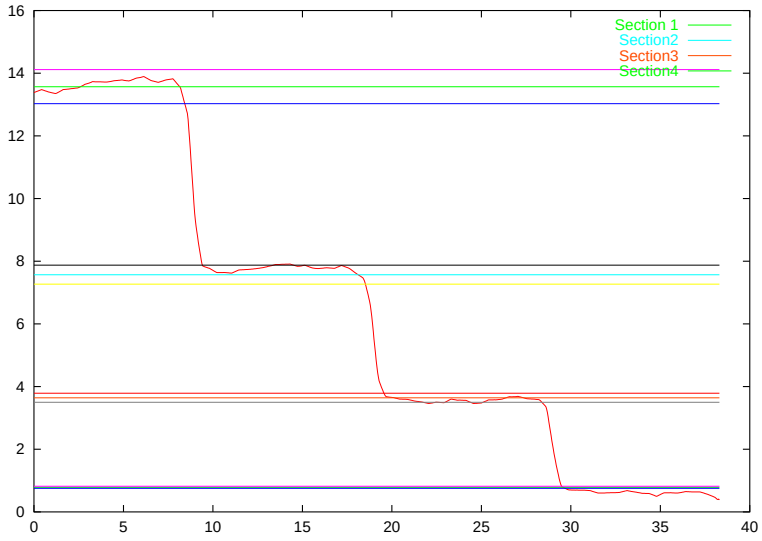


Figure 7.11. Section measurements along the segmented phantom of figure 7.10: It represents the values of the section across the trajectory extracted, with the deviation of 2% superimposed.